View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Theoretical Computer Science 15 (1981) 41–50
North-Holland Publishing Company

BOUNDED QUERY MACHINES: ON NP( ) AND NPQUERY( )*

Ronald V. BOOK and Celia WRATHALL

Department of Mathematics, University of California at Santa Barbara, Santa Barbara, CA 93106, U.S.A.

Communicated by M. Paterson
Received December 1979
Revised April 1980

Abstract. The NP( ) and NPQUERY( ) operators are studied in order to develop necessary and sufficient conditions for the class PSPACE to be equal to the union of the polynomial-time hierarchy.

1. Introduction

Is NP equal to PSPACE? If not, is the union of the polynomial-time hierarchy equal to PSPACE? The main result of this paper is a collection of necessary and sufficient conditions for the answer to the second question to be 'yes'.

The question of whether the class NP of languages accepted in polynomial time by nondeterministic machines is equal to the class PSPACE of languages accepted by machines that use at most polynomial work space has stimulated the investigation of the existence of a nontrivial polynomial-time hierarchy [3, 10, 11, 14]. The study of relativizations of NP and PSPACE has led to the investigation of NP( ) and PSPACE( ) as operators and it is known that PSPACE( ) is an idempotent operator (i.e., for all A, PSPACE(PSPACE(A)) = PSPACE(A)) while NP( ) is not idempotent (i.e., there exists an oracle set B such that NP(B) ≠ NP(NP(B)) [2, 3]).

A new class was introduced in [5]. For any oracle set A, let NPQUERY(A) (PQUERY(A)) be the class of languages accepted by nondeterministic (resp., deterministic) oracle machines that use A as oracle set, that use at most a polynomial amount of work space, and that are allowed to query the oracle only a polynomial number of times in any accepting computation. Theorem 3.1 of [5] states that NP = PSPACE if and only if for all oracle sets A, NP(A) = NPQUERY(A), and so one might try to resolve the question of NP = ? PSPACE by finding an oracle set B such that NP(B) ≠ NPQUERY(B) or by showing that for all A, NP(A) = NPQUERY(A). Many of the results in [5] make it appear that classes of the form

* This research was supported in part by the National Science Foundation under Grant MCS77-11360.

0304-3975/81/0000-0000/$02.50 © North-Holland Publishing Company

PQUERY(A) or NPQUERY(A) have structural properties that are very similar to those of P and NP and thus appear to behave more like classes specified by time-bounded machines than like classes specified by space-bounded machines. These facts shed light on the nature of relativized computation by restricted machines and in addition suggest that other similarities between PQUERY( ) and NPQUERY( ) on the one hand, and P( ) and NP( ) on the other hand, be investigated.

In the present paper the operators NP( ) and NPQUERY( ) are investigated in connection with the question of whether PSPACE is equal to the union of the polynomial-time hierarchy, that is, whether PSPACE is equal to the class of extended rudimentary languages. The most important single result (Theorem 5.5) is a collection of necessary and sufficient conditions for this to occur. To establish this result it is necessary to define 'the polynomial-query hierarchy relative to A', for an oracle set A; the definition of this hierarchy is similar to that of the polynomial-time hierarchy relative to an oracle set, and for each oracle set the two hierarchies relative to that set are very closely related. The fact that for every oracle set A, NP(NPQUERY(A)) = NPQUERY(NPQUERY(A)) (Corollary 3.5) is a useful tool in this development, in particular in comparing the polynomial-time and polynomial-query hierarchies relative to an oracle set and characterizing the polynomial-query hierarchy relative to an oracle set (Theorem 4.4).

The reason for introducing the polynomial-query classes and the polynomial-query hierarchy is not to define new classes but rather to provide tools to study (as in [5]) whether NP is equal to PSPACE or (in the present paper) whether PSPACE is equal to the union of the polynomial-time hierarchy. It is hoped that this approach will clarify insights gained from [2, 3, 10].

The results presented here add to our understanding of complexity-bounded operators as well as illustrating that requiring a bound on the number of times a machine performs an operation forces a class of languages specified by space-bounded machines to have properties very similar to those of a class specified by time-bounded machines. The proof techniques illustrate the power and usefulness of the algebraic approach to formal language theory for studying questions about computational complexity.

2. Notation

This paper is a sequel to [5] and it is assumed that the reader is familiar with the notation used there. Certain additional notions are explained here.

If $\mathscr{L}$ is a class of languages, then let co-$\mathscr{L}$ be the class of complements of $\mathscr{L}$, i.e., co-$\mathscr{L} = \{\Sigma^* - L \mid L$ is in $\mathscr{L}$ and Σ is a finite alphabet such that $L \subseteq \Sigma^*\}$.

If $\mathscr{L}_1$ and $\mathscr{L}_2$ are classes of languages, then let

$$\mathscr{L}_1 \wedge \mathscr{L}_2 = \{L_1 \cap L_2 \mid L_1 \in \mathscr{L}_1 \text{ and } L_2 \in \mathscr{L}_2\}.$$

If $\mathscr{L}$ is a class of languages, then

$$H_{\text{poly}}(\mathscr{L}) = \{h(L) \mid L \in \mathscr{L} \text{ and } h \text{ is polynomial-erasing on } L\}.$$

3. Representations

For any class $\mathscr{L}$ of languages, let $P(\mathscr{L}) = \bigcup\{P(A) \mid A \in \mathscr{L}\}$, $\text{NP}(\mathscr{L}) = \bigcup\{\text{NP}(A) \mid A \in \mathscr{L}\}$, $\text{PQUERY}(\mathscr{L}) = \bigcup\{\text{PQUERY}(A) \mid A \in \mathscr{L}\}$, $\text{NPQUERY}(\mathscr{L}) = \bigcup\{\text{NPQUERY}(A) \mid A \in \mathscr{L}\}$ and $\text{PSPACE}(\mathscr{L}) = \bigcup\{\text{PSPACE}(A) \mid A \in \mathscr{L}\}$. It is known that for all $\mathscr{L}$, $P(P(\mathscr{L})) = P(\mathscr{L})$ and $\text{PSPACE}(\text{PSPACE}(\mathscr{L})) = \text{PSPACE}(\mathscr{L})$, and it is easy to show that $\text{PQUERY}(\text{PQUERY}(\mathscr{L})) = \text{PQUERY}(\mathscr{L})$. In this section we represent classes of the form $\text{NPQUERY}(\mathscr{L})$ for certain classes $\mathscr{L}$.

Using the techniques of Ginsburg, Greibach, and Hopcroft [7] and of Wrathall [13], we can establish the following facts.

Lemma 3.1. (a) *If $\mathscr{L}$ is a class of languages that is closed under union with regular sets and marked concatenation (or marked union), then each of $P(\mathscr{L})$, $\text{NP}(\mathscr{L})$, $\text{PQUERY}(\mathscr{L})$, $\text{NPQUERY}(\mathscr{L})$ and $\text{PSPACE}(\mathscr{L})$ is closed under intersection, union and concatenation.*

(b) *If $\mathscr{L}$ is closed under concatenation (by regular sets), union with regular sets, inverse homomorphism, and Kleene*, then for each $A \in \mathscr{L}$ and homomorphism h, $h^{-1}((A \oplus \bar{A})^*)$ is in $\mathscr{L} \wedge \text{co-}\mathscr{L}$, and $\mathscr{L} \wedge \text{co-}\mathscr{L}$ is closed under inverse homomorphism.*

(c) *For any class $\mathscr{L}_1$ of languages, if $\mathscr{L}_2$ is $\text{NP}(\mathscr{L}_1)$ or $\text{NPQUERY}(\mathscr{L}_1)$, then $\mathscr{L}_2 \subseteq H_{\text{poly}}(\text{co-}\mathscr{L}_2)$.*

The representation of $\text{NPQUERY}(\mathscr{L})$ is the next result.

Theorem 3.2. *If $\mathscr{L}$ is a class of languages that is closed under inverse homomorphism, union with regular sets, concatenation, concatenation with regular sets, and Kleene*, then* $\text{NPQUERY}(\mathscr{L}) = H_{\text{poly}}(\mathscr{L} \wedge \text{co-}\mathscr{L} \wedge \text{DSPACE(LIN)})$.

Proof. It is clear that $\text{NPQUERY}(\mathscr{L})$ is closed under polynomial-erasing homomorphism and intersection, and that each of $\mathscr{L}$, co-$\mathscr{L}$, and DSPACE(LIN) are subclasses of $\text{NPQUERY}(\mathscr{L})$. Thus,

$$H_{\text{poly}}(\mathscr{L} \wedge \text{co-}\mathscr{L} \wedge \text{DSPACE(LIN)}) \subseteq \text{NPQUERY}(\mathscr{L}).$$

If L is a language in $\text{NPQUERY}(\mathscr{L})$, then there is some set A such that $L \in \text{NPQUERY}(A)$. By Lemma 3.2 of [5], there exist homomorphisms h_1 and h_2 and a language $L_1 \in \text{NSPACE(LIN)}$ such that $h_1(L_1 \cap h_2^{-1}((A \oplus \bar{A})^*)) = L$ and h_1 is polynomial-erasing on $L_1 \cap h_2^{-1}((A \oplus \bar{A})^*)$. From Lemma 3.1(b) we see that $h_2^{-1}((A \oplus \bar{A})^*)$ is in $\mathscr{L} \wedge \text{co-}\mathscr{L}$ and so L is in $H_{\text{poly}}(\text{NSPACE(LIN)} \wedge \mathscr{L} \wedge \text{co-}\mathscr{L})$. Since L was chosen arbitrarily in $\text{NPQUERY}(\mathscr{L})$, this shows that $\text{NPQUERY}(\mathscr{L}) \subseteq$

$H_{\text{poly}}(\mathscr{L} \wedge \text{co-}\mathscr{L} \wedge \text{NSPACE(LIN)})$. But $\text{NSPACE(LIN)} \subseteq H_{\text{poly}}(\text{DSPACE(LIN)})$ and $\mathscr{L} \wedge \text{co-}\mathscr{L}$ is closed under inverse homomorphism so that

$$H_{\text{poly}}(\mathscr{L} \wedge \text{co-}\mathscr{L} \wedge \text{DSPACE(LIN)}) = H_{\text{poly}}(\mathscr{L} \wedge \text{co-}\mathscr{L} \wedge \text{NSPACE(LIN)})$$

and so

$$\text{NPQUERY}(\mathscr{L}) \subseteq H_{\text{poly}}(\mathscr{L} \wedge \text{co-}\mathscr{L} \wedge \text{DSPACE(LIN)}).$$

Using an argument similar to the proof of Theorem 3.2, one can obtain the following fact.

Theorem 3.3. *If $\mathscr{L}$ is a class of languages that is closed under inverse homomorphism, union with regular sets, concatenation, concatenation with regular sets, and Kleene*, then*

$$\begin{aligned}\text{NP}(\mathscr{L}) &= \{h(L) \mid L \in \mathscr{L} \wedge \text{co-}\mathscr{L} \wedge \text{DTIME(LIN)} \\ &\qquad \textit{and } h \textit{ is polynomial-erasing on } L\} \\ &= H_{\text{poly}}(\mathscr{L} \wedge \text{co-}\mathscr{L} \wedge \text{DTIME(LIN)}).\end{aligned}$$

The following fact is useful.

Corollary 3.4. *Let $\mathscr{L}_1$ be a singleton class or a class closed under marked union.*
(a) *If $\mathscr{L}_2 = \text{NP}(\mathscr{L}_1)$, then* $\text{NPQUERY}(\mathscr{L}_2) = H_{\text{poly}}(\text{co-}\mathscr{L}_2 \wedge \text{DSPACE(LIN)})$.
(b) *If $\mathscr{L}_3 = \text{NPQUERY}(\mathscr{L}_1)$, then* $\text{NPQUERY}(\mathscr{L}_3) = H_{\text{poly}}(\text{co-}\mathscr{L}_3) = \text{NP}(\mathscr{L}_3)$.

Proof. Under the assumptions on $\mathscr{L}_1$, both $\text{NP}(\mathscr{L}_1)$ and $\text{NPQUERY}(\mathscr{L}_1)$ are closed under inverse homomorphism, union, concatenation, and Kleene *, and contain all regular sets.

(a) From Theorem 3.2 we see that

$$\text{NPQUERY}(\mathscr{L}_2) = H_{\text{poly}}(\mathscr{L}_2 \wedge \text{co-}\mathscr{L}_2 \wedge \text{DSPACE(LIN)}).$$

Since $\mathscr{L}_2 = \text{NP}(\mathscr{L}_1)$, $\mathscr{L}_2 \subseteq H_{\text{poly}}(\text{co-}\mathscr{L}_2)$ by Lemma 3.1(c). Thus,

$$\mathscr{L}_2 \wedge \text{co-}\mathscr{L}_2 \subseteq H_{\text{poly}}(\text{co-}\mathscr{L}_2) \wedge \text{co-}\mathscr{L}_2 = H_{\text{poly}}(\text{co-}\mathscr{L}_2)$$

since $\text{co-}\mathscr{L}_2$ is an intersection-closed preAFL. Therefore

$$\begin{aligned}\text{NPQUERY}(\mathscr{L}_2) &\subseteq H_{\text{poly}}(H_{\text{poly}}(\text{co-}\mathscr{L}_2) \wedge \text{DSPACE(LIN)}) \\ &= H_{\text{poly}}(\text{co-}\mathscr{L}_2 \wedge \text{DSPACE(LIN)}) \\ &\subseteq H_{\text{poly}}(\mathscr{L}_2 \wedge \text{co-}\mathscr{L}_2 \wedge \text{DSPACE(LIN)}) \\ &= \text{NPQUERY}(\mathscr{L}_2),\end{aligned}$$

so that $\text{NPQUERY}(\mathscr{L}_2) = H_{\text{poly}}(\text{co-}\mathscr{L}_2 \wedge \text{DSPACE(LIN)})$ as claimed.

(b) From Theorem 3.2 we see that

$$\text{NPQUERY}(\mathscr{L}_3) = H_{\text{poly}}(\mathscr{L}_3 \wedge \text{co-}\mathscr{L}_3 \wedge \text{DSPACE(LIN)})$$
$$\subseteq H_{\text{poly}}(\text{co-}\mathscr{L}_3 \wedge \text{DSPACE(LIN)}).$$

But

$$\text{DSPACE(LIN)} \subseteq \text{PSPACE} = \text{co-PSPACE} \subseteq \text{NPQUERY}(\mathscr{L}_1) = \mathscr{L}_3$$

so that $\text{DSPACE(LIN)} \subseteq \text{co-PSPACE} \subseteq \text{co-}\mathscr{L}_3$. Now co-$\mathscr{L}_3$ is closed under intersection and so $\text{NPQUERY}(\mathscr{L}_3) \subseteq H_{\text{poly}}(\text{co-}\mathscr{L}_3)$ and $H_{\text{poly}}(\text{co-}\mathscr{L}_3) \subseteq \text{NPQUERY}(\mathscr{L}_3)$ so that $\text{NPQUERY}(\mathscr{L}_3) = H_{\text{poly}}(\text{co-}\mathscr{L}_3)$. Similarly,

$$\text{NP}(\mathscr{L}_3) = H_{\text{poly}}(\mathscr{L}_3 \wedge \text{co-}\mathscr{L}_3 \wedge \text{DTIME(LIN)}) = H_{\text{poly}}(\text{co-}\mathscr{L}_3).$$

An interesting question arises from consideration of Corollary 3.4. Let $\mathscr{L}$ be a class of languages such that $\text{DTIME(LIN)} \subseteq \mathscr{L} \subseteq \text{DSPACE(LIN)}$ and $\mathscr{L}$ is closed under marked union. Let RUD be the class of rudimentary languages so that RUD is the smallest Boolean-closed AFL containing the language $\{a^n b^n \mid n > 0\}$ [15]. What is the class $H_{\text{poly}}(\mathscr{L} \wedge \text{co-}\mathscr{L} \wedge \text{RUD})$? This class contains all of the extended rudimentary languages since the latter class is equal to $H_{\text{poly}}(\text{RUD})$ and this class is contained in PSPACE since $\mathscr{L} \wedge \text{co-}\mathscr{L} \wedge \text{RUD}$ is a subclass of DSPACE(LIN) and $H_{\text{poly}}(\text{PSPACE}) = \text{PSPACE}$.

The next fact follows from Corollary 3.4.

Corollary 3.5. *For any set A,* $\text{NPQUERY}(\text{NPQUERY}(A)) = \text{NP}(\text{NPQUERY}(A))$.

Another useful fact follows from Theorem 3.2.

Corollary 3.6. *If $\mathscr{L}$ is a class of languages that is closed under inverse homomorphism, union with regular sets, concatenation, concatenation with regular sets, and Kleene *, and if* $\text{DSPACE(LIN)} \subseteq \mathscr{L}$, *then* $\text{NPQUERY}(\mathscr{L}) = \text{NP}(\mathscr{L})$.

4. Iterations of NPQUERY( )

In this section we review the notion of the polynomial-time hierarchy and its relativized versions and introduce a comparable hierarchy based on the NPQUERY( ) operator.

Let $\mathscr{L}$ be a class of languages. Let $\Sigma_1^{\text{P}}(\mathscr{L}) = \text{NP}(\mathscr{L})$ and for each $i \geq 1$, let $\Sigma_{i+1}^{\text{P}}(\mathscr{L}) = \text{NP}(\Sigma_i^{\text{P}}(\mathscr{L}))$. Let $\text{NP}^*(\mathscr{L}) = \bigcup_i \Sigma_i^{\text{P}}(\mathscr{L})$. For each $i \geq 1$, let $\Pi_i^{\text{P}}(\mathscr{L}) = \text{co-}\Sigma_i^{\text{P}}(\mathscr{L})$. If $\mathscr{L}$ is a singleton class, say $\mathscr{L} = \{A\}$, then write $\Sigma_i^{\text{P}}(A)$, $\text{NP}^*(A)$ and $\Pi_i^{\text{P}}(A)$ instead of $\Sigma_i^{\text{P}}(\{A\})$, $\text{NP}^*(\{A\})$ and $\Pi_i^{\text{P}}(\{A\})$, respectively. If $\mathscr{L} = \{\emptyset\}$ then write Σ_i^{P}, NP^*, and Π_i^{P} instead of $\Sigma_i^{\text{P}}(\emptyset)$, $\text{NP}^*(\emptyset)$, and $\Pi_i^{\text{P}}(\emptyset)$, respectively.

The structure $\Sigma_1^P \subseteq \Sigma_2^P \subseteq \cdots$ is known as the *polynomial-time hierarchy* even though it is not known whether it is a proper hierarchy. In fact, it is not known if any of the inclusions $\Sigma_k^P \subseteq \Sigma_{k+1}^P$ is proper; also, if P = NP, then for every k, $\Sigma_k^P = P$, [3, 11, 13, 14]. It is known that for each $i > 0$, $\Sigma_{i+1}^P = H_{poly}(\Pi_i^P)$ and that NP* is the class of extended rudimentary languages, the smallest class of languages containing the language $\{a^n b^n \mid n \geq 0\}$ and closed under the Boolean operations, inverse homomorphism, and polynomial-erasing homomorphism [13].

If A is an oracle set, the structure $\Sigma_1^P(A) \subseteq \Sigma_2^P(A) \subseteq \cdots$ is known as the *polynomial-time hierarchy relative to A*. Baker and Selman [3] have shown that there is an oracle set A such that $\Sigma_1^P(A) \subsetneq \Sigma_2^P(A) \subsetneq \Sigma_3^P(A)$ (see also [1]). It is known that for each $i > 0$, $\Sigma_{i+1}^P(A) = H_{poly}(\Pi_i^P(A))$ and that NP*(A) is the class of languages that are extended rudimentary in A, the smallest class of languages containing A and the language $\{a^n b^n \mid n \geq 0\}$ and closed under the Boolean operations, inverse homomorphism and polynomial-erasing homomorphism [13].

For every class $\mathcal{L}$ of languages, NP* $\subseteq$ NP*$(\mathcal{L}) \subseteq$ PSPACE$(\mathcal{L})$. If NP* = PSPACE, then the polynomial hierarchy is finite, and if NP*(A) = PSPACE(A) for some A, then the polynomial hierarchy relative to A is finite [13, 14]. For any class $\mathcal{L}$ of languages, NP*$(\mathcal{L})$ = PSPACE$(\mathcal{L})$ if and only if NP*$(\mathcal{L})$ is closed under a language-theoretic operation related to the transitive closure of length-preserving binary string relations [4].

Now consider the operator NPQUERY( ). Let $\mathcal{L}$ be a class of languages. Let $\Sigma_1^{PQ}(\mathcal{L})$ = NPQUERY$(\mathcal{L})$ and for each $i \geq 1$, let $\Sigma_{i+1}^{PQ}(\mathcal{L})$ = NPQUERY$(\Sigma_i^{PQ}(\mathcal{L}))$. Let NPQUERY*$(\mathcal{L}) = \bigcup_i \Sigma_i^{PQ}(\mathcal{L})$. For each $i \geq 1$, let $\Pi_i^{PQ}(\mathcal{L})$ = co-$\Sigma_i^{PQ}(\mathcal{L})$. If $\mathcal{L}$ is a singleton class, say $\mathcal{L} = \{A\}$, then write $\Sigma_i^{PQ}(A)$, NPQUERY*(A) and $\Pi_i^{PQ}(A)$ instead of $\Sigma_i^{PQ}(\{A\})$, NPQUERY*$(\{A\})$ and $\Pi_i^{PQ}(\{A\})$, respectively.

If A is an oracle set, the structure $\Sigma_1^{PQ}(A) \subseteq \Sigma_2^{PQ}(A) \subseteq \cdots$ will be called the *polynomial-query hierarchy relative to A*. The argument of Baker and Selman [3] can be used to show that there is an oracle set A such that $\Sigma_1^{PQ}(A) \subsetneq \Sigma_2^{PQ}(A) \subsetneq \Sigma_3^{PQ}(A)$. We will show that for all A and all $i > 0$, $\Sigma_{i+1}^{PQ}(A) = H_{poly}(\Pi_i^{PQ}(A)) = NP(\Sigma_i^{PQ}(A))$.

Theorem 4.1. *Let $\mathcal{L}$ be a singleton class or a class closed under marked union. For each $i > 0$, $\Sigma_{i+1}^{PQ}(\mathcal{L}) = H_{poly}(\Pi_i^{PQ}(\mathcal{L})) = NP(\Sigma_i^{PQ}(\mathcal{L}))$.*

Proof. Since $\Sigma_1^{PQ}(\mathcal{L})$ = NPQUERY$(\mathcal{L})$ and $\Pi_1^{PQ}(\mathcal{L})$ = co-$\Sigma_1^{PQ}(\mathcal{L})$, it follows from Corollary 3.4 that NPQUERY$(\Sigma_1^{PQ}(\mathcal{L})) = NP(\Sigma_1^{PQ}(\mathcal{L})) = H_{poly}(\Pi_1^{PQ}(\mathcal{L}))$. Since $\Sigma_2^{PQ}(\mathcal{L})$ is defined to be NPQUERY$(\Sigma_1^{PQ}(\mathcal{L}))$, the result is true for $i = 1$. The inductive step also follows from Corollary 3.4 when we note that for every class $\mathcal{C}$, both NP$(\mathcal{C})$ and NPQUERY$(\mathcal{C})$ are closed under marked union.

Corollary 4.2. *For any oracle set A and every $i > 0$, $\Sigma_{i+1}^{PQ}(A) = H_{poly}(\Pi_i^{PQ}(A)) = NP(\Sigma_i^{PQ}(A))$.*

Corollary 4.3. *For any oracle set A,* NPQUERY*(A) = NP*(NPQUERY(A)) = NP*$(\Sigma_1^{PQ}(A))$.

Now we can obtain a characterization theorem for classes of the form NPQUERY*(A).

Theorem 4.4. *For every oracle set A, the class* NPQUERY*(A) *is the smallest class of languages that contains A and every language in* DSPACE(LIN) *and that is closed under the Boolean operations, inverse homomorphism and polynomial-erasing homomorphism.*

Proof. Let $\mathscr{L}(A)$ be the smallest class of languages that contains A and every language in DSPACE(LIN) and that is closed under the Boolean operations, inverse homomorphism, and polynomial-erasing homomorphism. Using the results in Section 3, we see that for each $i>0$ the class $\Sigma_i^{PQ}(A)$ is closed under union, intersection, inverse homomorphism and polynomial-erasing homomorphism, and clearly $A\in\Sigma_i^{PQ}(A)$ and $\text{DSPACE(LIN)}\subseteq\text{PSPACE}=\Sigma_1^{PQ}(\emptyset)\subseteq\Sigma_i^{PQ}(A)$. Thus $\text{NPQUERY*}(A)=\bigcup_i\Sigma_i^{PQ}(A)$ is closed under these operations, $A\in\text{NPQUERY*}(A)$, and $\text{DSPACE(LIN)}\subseteq\text{NPQUERY*}(A)$. Since for each $i>0$, $\Pi_i^{PQ}(A)=\text{co-}\Sigma_i^{PQ}(A)$ and $\Sigma_{i+1}^{PQ}(A)=H_{\text{poly}}(\Pi_i^{PQ}(A))\supseteq\Pi_i^{PQ}(A)$ (Corollary 4.2), we see that NPQUERY*(A) is closed under complementation. Thus, $\mathscr{L}(A)\subseteq$ NPQUERY*(A) since $\mathscr{L}(A)$ is the smallest class with these properties.

Recall from Lemma 3.3 of [5] that NPQUERY(A) is the smallest class containing DSPACE(LIN) and every language of the form $h^{-1}((A\oplus\bar{A})^*)$ where h is a homomorphism, and closed under intersection and polynomial-erasing homomorphism. Thus, $\Sigma_1^{PQ}(A)=\text{NPQUERY}(A)\subseteq\mathscr{L}(A)$. Suppose that for some $i>0$, $\Sigma_i^{PQ}(A)\subseteq\mathscr{L}(A)$. Then $\Pi_i^{PQ}(A)\subseteq\mathscr{L}(A)$ since $\mathscr{L}(A)$ is closed under complementation. Since $\Sigma_{i+1}^{PQ}(A)=H_{\text{poly}}(\Pi_i^{PQ}(A))$ (Corollary 4.2) and $\mathscr{L}(A)$ is closed under polynomial-erasing homomorphism, we have $\Sigma_{i+1}^{PQ}(A)\subseteq\mathscr{L}(A)$. Thus, by induction, $\Sigma_i^{PQ}(A)\subseteq\mathscr{L}(A)$ for every i, and so $\text{NP*}(A)=\bigcup_i\Sigma_i^{PQ}(A)\subseteq\mathscr{L}(A)$.

Corollary 4.5. *There exists a language L_0 such that for every oracle set A,* NPQUERY*(A) *is the smallest class of languages that contains A and L_0 and is closed under the Boolean operations, inverse homomorphism, and polynomial-erasing homomorphism.*

Proof. This follows from Theorem 4.4 when one notices that the AFL generator L_0 for DSPACE(LIN) described by Wegbreit [12] has the property that for every $L\in\text{DSPACE(LIN)}$ there is a homomorphism h such that $h^{-1}(L_0)=L-\{e\}$.

5. NP* = ?PSPACE

The principal reason to compare the operators NP( ) and NPQUERY( ) is the question of whether NP* is equal to PSPACE. It is possible that $\text{NP}\subsetneq\text{PSPACE}$ but NP* = PSPACE so that for some i, $\Sigma_i^P=\text{PSPACE}$. This situation is investigated in the following sequence of results.

Theorem 5.1. *For each $i \geq 1$ the following are equivalent:*

(a) $\Sigma_i^P = \text{PSPACE}$;

(b) *for every oracle set* A, $\text{DSPACE(LIN)} \subseteq \Sigma_i^P(A)$;

(c) *for every oracle set* A, $\text{NPQUERY}(A)$ *is the smallest class of languages that contains every language in* Σ_i^P *and every language* $h^{-1}((A \oplus \bar{A})^*)$ *where* h *is a homomorphism and that is closed under intersection and polynomial-erasing homomorphism.*

Proof. (a)$\Rightarrow$(c). From Lemma 3.3 of [5], we know that for every oracle set A, $\text{NPQUERY}(A)$ is the smallest class of languages that contains every language in DSPACE(LIN) and every language $h^{-1}((A \oplus \bar{A})^*)$ where h is a homomorphism and that is closed under intersection and polynomial-erasing homomorphism. Since $\text{DSPACE(LIN)} \subsetneq \text{PSPACE} \subseteq \text{NPQUERY}(A)$, if $\Sigma_i^P = \text{PSPACE}$, then $\text{NPQUERY}(A)$ is precisely the class specified in (c).

(c)$\Rightarrow$(b). The class $\Sigma_i^P(A)$ is closed under intersection and polynomial-erasing homomorphism, and every language of the form $h^{-1}((A \oplus \bar{A})^*)$ is in $\Sigma_1^P(A)$ and hence in $\Sigma_i^P(A)$; also, $\Sigma_i^P \subseteq \Sigma_i^P(A)$. Thus, if (c) is true, then $\text{NPQUERY}(A) \subseteq \Sigma_i^P(A)$. Since $\text{DSPACE(LIN)} \subsetneq \text{PSPACE} \subseteq \text{NPQUERY}(A)$, we see that $\text{DSPACE(LIN)} \subsetneq \Sigma_i^P(A)$.

(b)$\Rightarrow$(a). If (b) is true, then choosing $A = \emptyset$ yields $\text{DSPACE(LIN)} \subseteq \Sigma_i^P$. But $\text{PSPACE} = H_{\text{poly}}(\text{DSPACE(LIN)})$ and $H_{\text{poly}}(\Sigma_i^P) = \Sigma_i^P$. Since $\Sigma_i^P \subseteq \text{PSPACE}$, this means that $\Sigma_i^P = \text{PSPACE}$.

Theorem 5.2. *For every oracle set* A *the following are equivalent:*

(a) *there exists an integer* i *such that* $\text{DSPACE(LIN)} \subseteq \Sigma_i^P(A)$;

(b) $\text{DSPACE(LIN)} \subseteq \text{NP}^*(A)$;

(c) $\text{NP}^*(A) = \text{NPQUERY}^*(A)$.

Proof. Obviously (a) implies (b), and (c) implies (b) since $\text{DSPACE(LIN)} \subseteq \text{NPQUERY}(\emptyset) \subseteq \text{NPQUERY}^*(A)$. If (b) is true, then let $L_0 \in \text{DSPACE(LIN)}$ be a language such that for every $L \in \text{DSPACE(LIN)}$ there is a homomorphism h such that $h^{-1}(L_0) = L - \{e\}$. (One such language L_0 is described in [12].) Since $L_0 \in \text{DSPACE(LIN)} \subseteq \text{NP}^*(A)$ there exists some index i such that $L_0 \in \Sigma_i^P(A)$, and since $\Sigma_i^P(A)$ is closed under inverse homomorphism and union with $\{e\}$ it follows that $\text{DSPACE(LIN)} \subseteq \Sigma_i^P(A)$ so (a) is true. Again if (b) is true, then consider $\text{NPQUERY}^*(A)$, which from Corollary 4.3 is equal to $\text{NP}^*(\text{NPQUERY}(A))$. Using Corollary 3.4 and the properties of $\text{NP}^*(\ )$, $\text{NPQUERY}(A) \subseteq \text{NPQUERY}(\text{NP}(A)) = H_{\text{poly}}(\text{co-NP}(A) \wedge \text{DSPACE(LIN)}) \subseteq \text{NP}^*(A)$. Therefore $\text{NPQUERY}^*(A) = \text{NP}^*(\text{NPQUERY}(A)) \subseteq \text{NP}^*(\text{NP}^*(A)) = \text{NP}^*(A)$ and (c) is true.

Corollary 5.3. *For every oracle set* A, *if there exists an integer* i *such that* $\text{DSPACE(LIN)} \subseteq \Sigma_i^P(A)$ *or, equivalently,* $\text{PSPACE} \subseteq \Sigma_i^P(A)$, *then for all* $j > 0$, $\Sigma_j^{PQ}(A) \subseteq \Sigma_{i+j-1}^P(A)$.

Notice that for any oracle set A the question '$\text{NP}^*(A) = ?\ \text{NPQUERY}^*(A)$' apparently compares two possibly infinite hierarchies. However Theorem 5.2 shows

that it is sufficient to find that DSPACE(LIN) $\subseteq \Sigma_i^P(A)$ (or, equivalently, $\Sigma_1^{PQ}(A) \subseteq \Sigma_i^P(A)$) at some finite point in the polynomial-query hierarchy relative to A.

For any time and space constructable function $f(n)$ and any set A, let DQUERY(f, A) (NQUERY(f, A)) be the class of all languages $L(M, A)$ where M is a deterministic (resp. nondeterministic) oracle machine that uses at most $f(n)$ work space and in any accepting computation is allowed to query its oracle at most $f(n)$ times.

Corollary 5.4. *For every oracle set A, every integer $k > 0$, and every integer $i > 0$, if $f(n) = n^k$, then $\Sigma_i^P(A) \neq$ DQUERY(f, A), $\Sigma_i^P(A) \neq$ NQUERY(f, A), NP*(A) $\neq$ DQUERY(f, A), and NP*(A) $\neq$ NQUERY(f, A).*

Proof. Each class $\Sigma_i^P(A)$ is closed under polynomial-erasing homomorphism as is NP*(A). The closure of NQUERY(f, A) or DQUERY(f, A) under polynomial-erasing homomorphism is NPQUERY(A). As noted in [5], DQUERY(f, A) $\subseteq$ NQUERY(f, A) $\subseteq$ NPQUERY(A). Thus the inequalities follow from Theorem 5.2.

From Theorems 5.1 and 5.2, we have the following fact.

Theorem 5.5. *The following are equivalent:*

(a) NP* = PSPACE;

(b) *there exists an integer i such that* Σ_i^P = PSPACE;

(c) *for every oracle set A there exists an integer i such that* DSPACE(LIN) $\subseteq \Sigma_i^P(A)$;

(d) *for every oracle set A,* DSPACE(LIN) $\subseteq$ NP*(A);

(e) *for every oracle set A,* NP*(A) = NPQUERY*(A).

(f) *for every oracle set A there exists an integer $i > 0$ such that* NPQUERY(A) *is the smallest class of languages that contains every language in Σ_i^P and every language $h^{-1}((A \oplus \bar{A})^*)$ where h is a homomorphism and that is closed under intersection and polynomial-erasing homomorphism.*

It is conceivable that there exists an integer i such that Σ_i^P = PSPACE but for some oracle set A the polynomial-time hierarchy relative to A is infinite. In such a case the polynomial-query hierarchy relative to A is also infinite since NP*(A) = NPQUERY*(A) (Theorem 5.5) and for all j, $\Sigma_j^P(A) \subseteq \Sigma_j^{PQ}(A)$. Similarly, if Σ_i^P = PSPACE and for some A the polynomial-query hierarchy relative to A is infinite, then the polynomial-time hierarchy relative to A is also infinite.

One can also compare classes such as NP*(A) and NPQUERY*(A) with PSPACE(A). It is easy to establish the following result.

Theorem 5.6. *For every oracle set A,* NP*(A) = PSPACE(A) *if and only if* DSPACE(LIN, A) $\subseteq$ NP*(A), *and* NPQUERY*(A) = PSPACE(A) *if and only if* DSPACE(LIN, A) $\subseteq$ NPQUERY*(A).

Corollary 5.7. *For every oracle set A and integer $k > 0$,* DSPACE(n^k, A) $\neq$ NPQUERY*(A), NSPACE(n^k, A) $\neq$ NPQUERY*(A), DSPACE(n^k, A) $\neq$ NP*(A), *and* NSPACE(n^k, A) $\neq$ NP*(A).

6. Remarks

From the results of [5] and the present paper, it is clear that classes of the form PQUERY(A) and NPQUERY(A) and operators of the form NPQUERY( ) and NPQUERY*( ) are natural objects to consider. Theorem 2.1 of [5] shows that NP = PSPACE if and only if for every oracle set A, NP(A) = NPQUERY(A), while Theorem 5.5 of the present paper shows that the union of the polynomial-time hierarchy is equal to PSPACE, i.e., NP* = PSPACE, if and only if for every oracle set A, NP*(A) = NPQUERY*(A).

Baker, Gill and Solovay [2], Baker and Selman [3] and Simon and Gill [10] separate pairs of relativized classes such as (P(A), NP(A)), (NP(A), PSPACE(A)) and ($\Sigma_i^P(A)$, PSPACE(A)). These separation results are obtained by using versions of the diagonalization techniques of elementary recursive function theory. It is now clear that to settle questions such as NP = ? PSPACE and NP* = ? PSPACE, the appropriate relativizations of PSPACE are classes of the form NPQUERY(A) and NPQUERY*(A) instead of PSPACE(A). It is not known whether the standard diagonalization techniques can be used to separate a pair of classes NP(A), NPQUERY(A) or a pair NP*(A), NPQUERY*(A).

References

[1] D. Angluin, On counting problems and the polynomial-time hierarchy, *Theoret. Comput. Sci.* **12** (1980) 161–173.

[2] T. Baker, J. Gill and R. Solovay, Relativizations of the P = ?NP question, *SIAM J. Comput.* **4** (1975) 431–442.

[3] T. Baker and A. Selman, A second step towards the polynomial time hierarchy, *Theoret. Comput. Sci.* **8** (1979), 177–187.

[4] R. Book, On languages accepted by space-bounded oracle machines, *Acta Informat.* **12** (1979) 177–185.

[5] R. Book, Bounded query machines: on NP and PSPACE, *Theoret. Comput. Sci.* **15** (1981) 27–39, this volume.

[6] R. Book, Time, space, and relativizations, in preparation.

[7] S. Ginsburg, S. Greibach and J. Hopcroft, Pre-AFL, *Studies in Abstract Families of Languages*, Memoir No. 87 (AMS, Providence, RI, 1969) 41–51.

[8] R. Ladner, N. Lynch and A. Selman, A comparison of polynomial time reducibilities, *Theoret. Comput. Sci.* **7** (1978) 185–195.

[9] D. Rosenkrantz and R. Stearns, A lattice of reducibilities, manuscript.

[10] I. Simon and J. Gill, Polynomial reducibilities and upwards diagonalizations, *Proc. 9th ACM Symposium on Theory of Computing* (1977) 186–194.

[11] L. Stockmeyer, The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1976) 1–22.

[12] B. Wegbreit, A generator of context-sensitive languages, *J. Comput. System Sci.* **3** (1969) 456–462.

[13] C. Wrathall, Subrecursive predicates and automata, Ph.D. dissertation, Harvard University (1975).

[14] C. Wrathall, Complete sets and the polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1976) 23–33.

[15] C. Wrathall, Rudimentary predicates and relative computation, *SIAM J. Comput.* **7** (1976), 194–209.